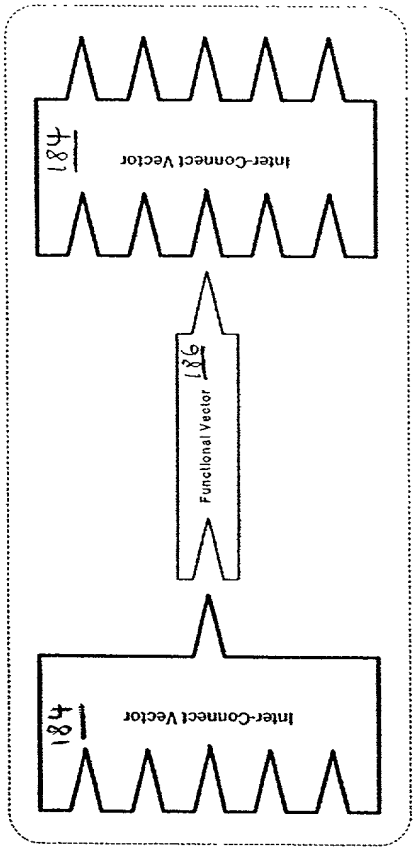

Figure 1A

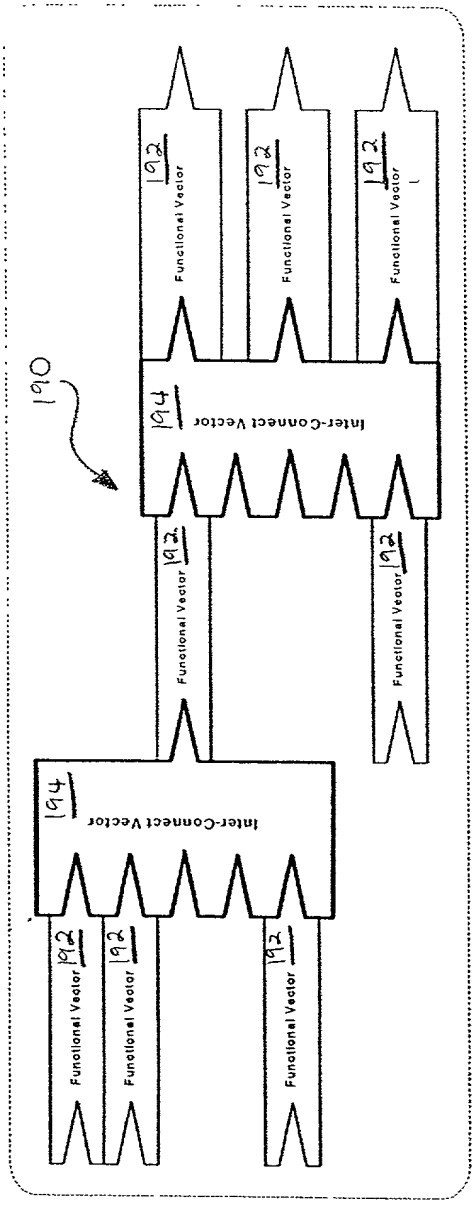

Figure 1B

005444.P004

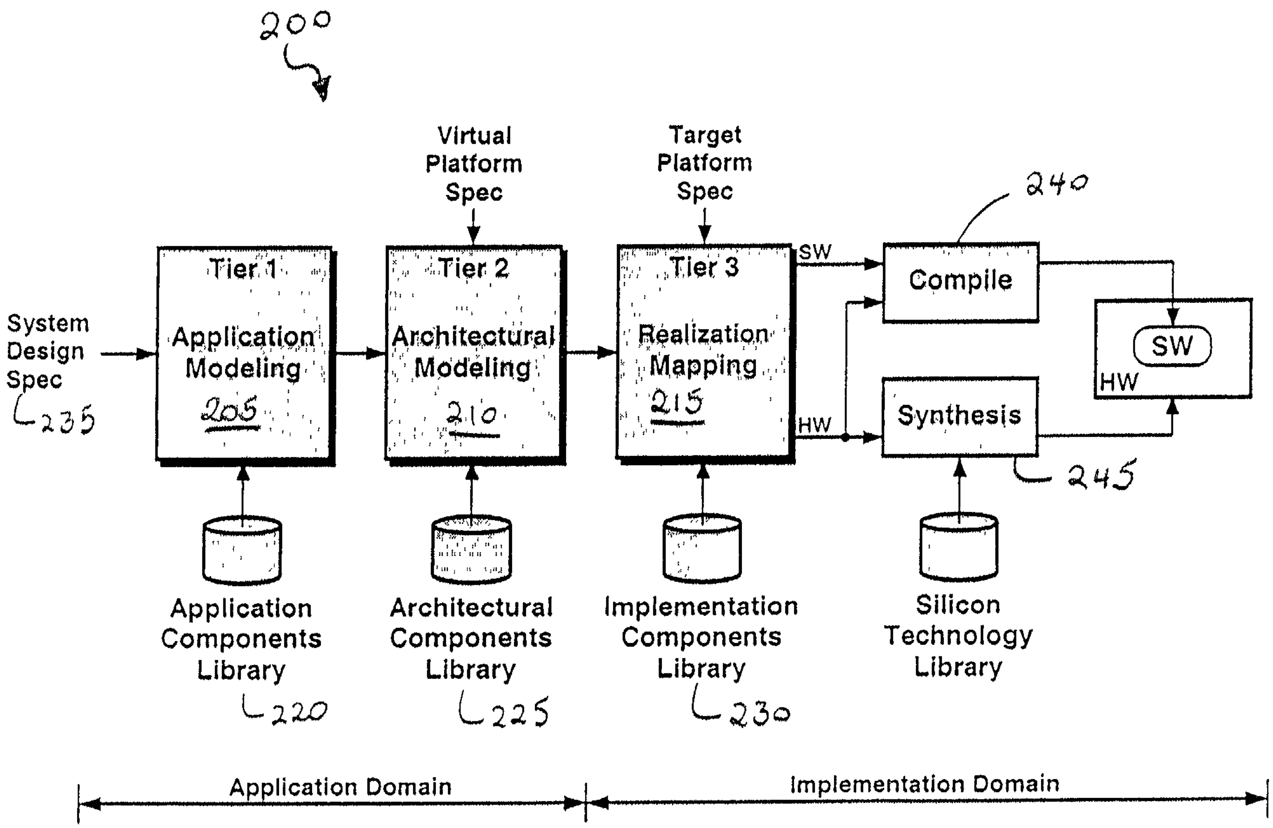

Figure 2

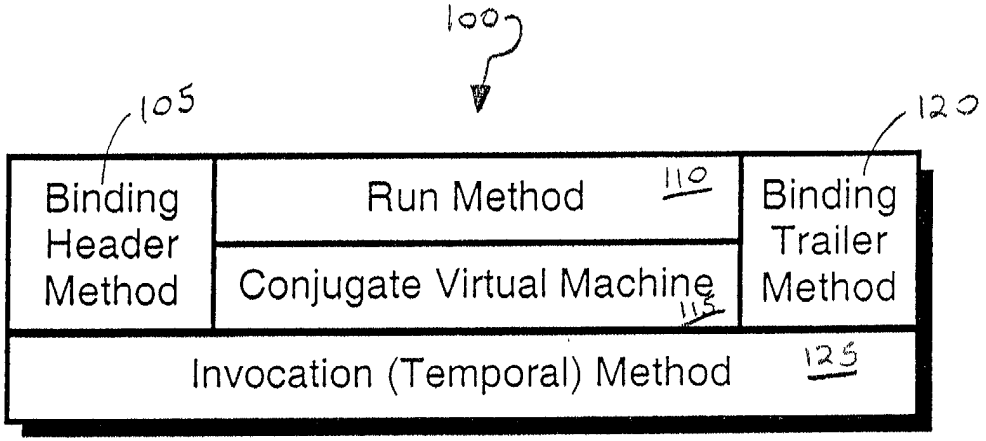

Figure 3A

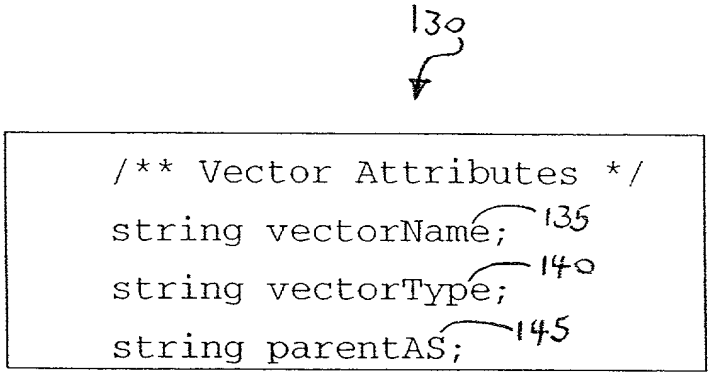

Figure 3B

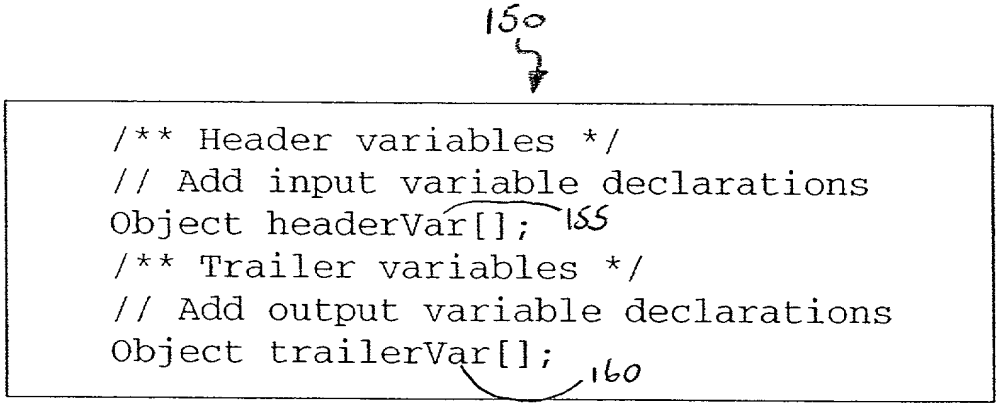

Figure 3C

170

172

```
    /** Vector Constructor Method: Construct an actor with the given vector name */
    public udmVectorClassName (string vectorName, udmVector inVector[], udmVector
outVector[])
    {   // Call constructor in base class
        super(vectorName, parentAS, inVector, outVector);
        // Perform any initialization that needs to be done in the constructor
    }

    /** This method contains the actual behavior of the vector  */
    private boolean vectorRun()                                          174
    {
        // Perform the vector processing
        return true;  // (or false if you want to terminate the thread)
    }

    /** This is the invocation method that checks to see if the vector is ready to run */
    private void vectorInvocation()                                      176
    {
        while ( !headerDataReady() ) vectorWait();
    }

    /** Get the header input data */
    private void getHeaderInput()                                        178
    {
        // Get input data from interconnect vector
        headerData = vectorGet();
    }

    /** Send the trailer output data  */
    private void sendTrailerOutput()                                     180
    {
        // Send output data to the interconnect vector
        vectorSend( trailerData );
    }

    /** run is the method that is started by Java when the thread is started */
    public void run()                                                    182
    {
        boolean runThread = true;

        // Initialize the vector
        initialize();
        while ( runThread )
        {
            // Call invocation method
            vectorInvocation();

            // Get input data
            getHeaderInput();

            // Do the processing for the vector
            runThread = vectorRun();

            // Send output data
            sendTrailerOutput();
        }

        // Perform final cleanup before vector thread exits
        wrapup();
    }
```

Figure 3D

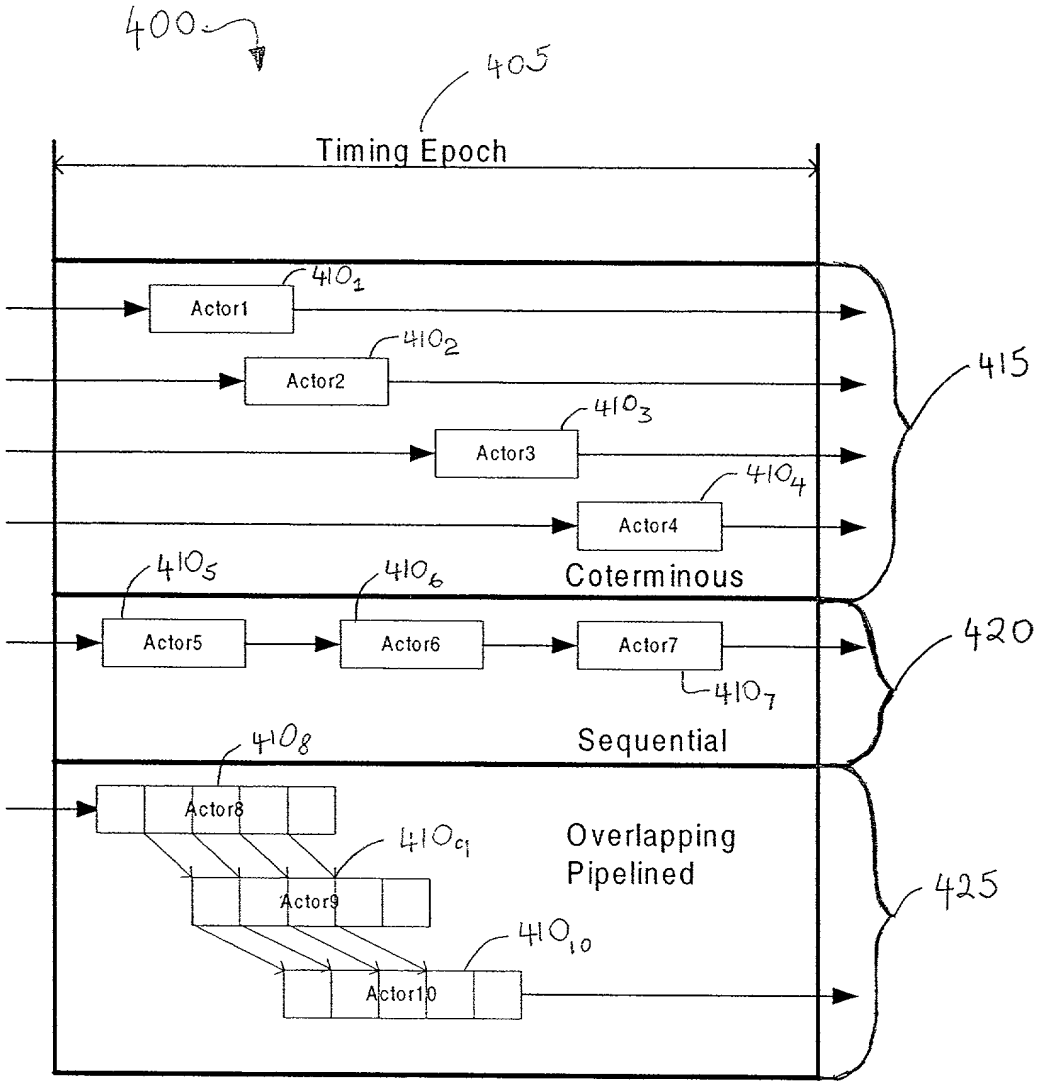
400
405
Timing Epoch
Actor1
410₁
Actor2
410₂
Actor3
410₃
Actor4
410₄
415
Coterminous
Actor5
410₅
Actor6
410₆
Actor7
410₇
420
Sequential
Actor8
410₈
Actor9
410c1
Actor10
410₁₀
Overlapping
Pipelined
425

Figure 4

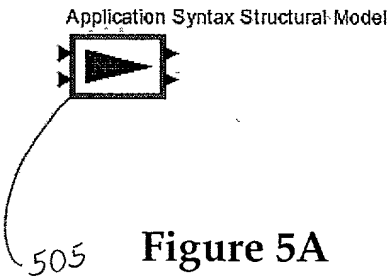

Figure 5A

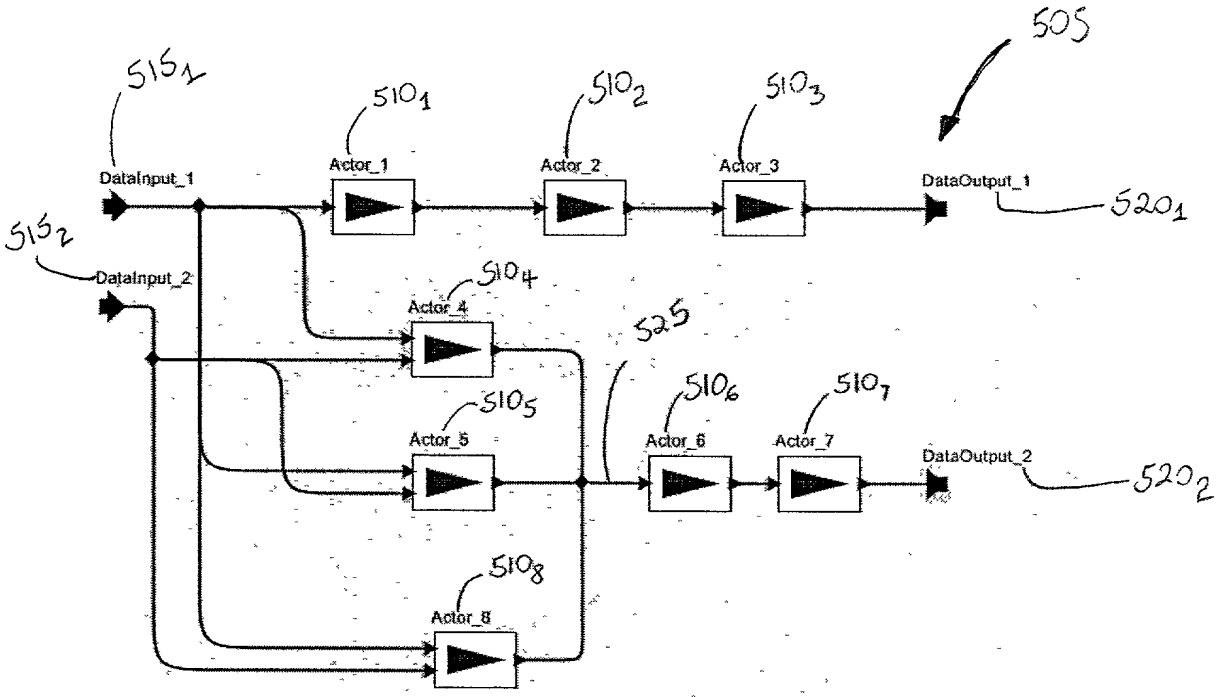

Figure 5B

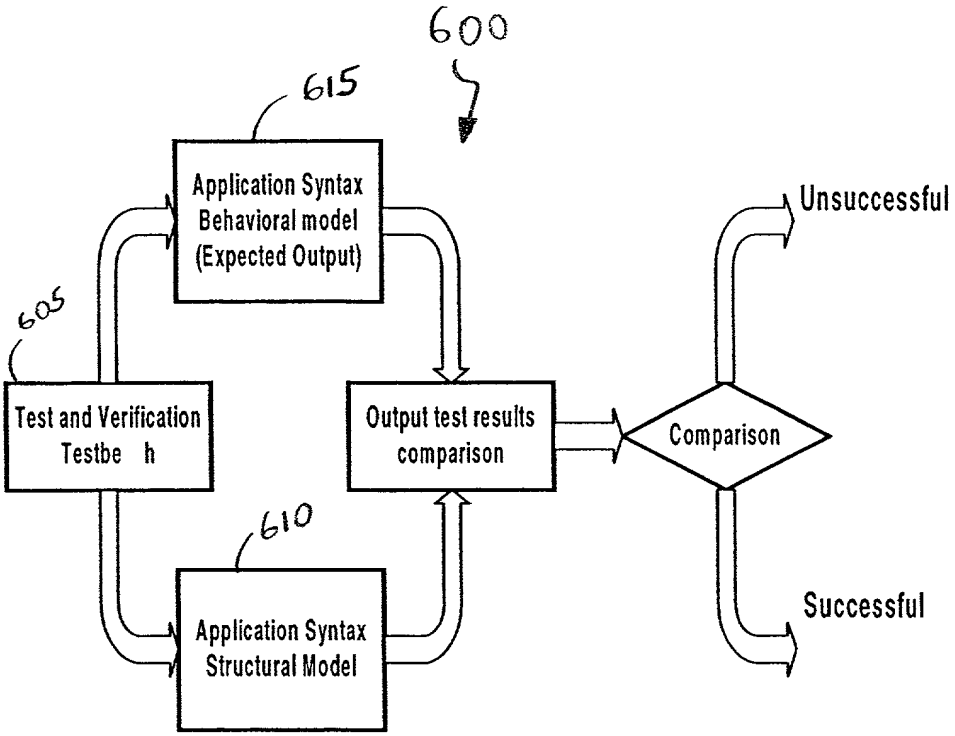
600
615
Application Syntax
Behavioral model
(Expected Output)
605
Test and Verification
Testbe h
Output test results
comparison
Comparison
Unsuccessful
Successful
610
Application Syntax
Structural Model

Figure 6

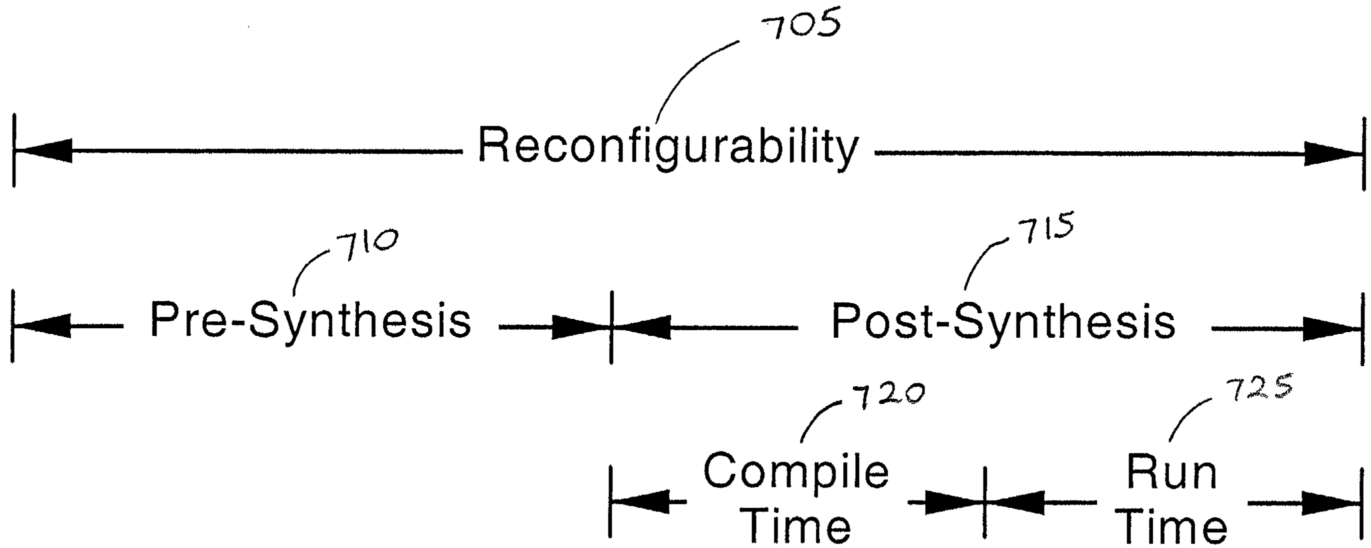

Figure 7

005444.P004